**K-Nearest Neighbor**

**Write a program in python to perform k-nearest neighbor classifier on on the “Iris” dataset available with Scikit Learn. Generate the Confusion Matrix and plot the graph of Mean Error against K-value.**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import warnings

warnings.filterwarnings('ignore')

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# Assign colum names to the dataset

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']

# Read dataset to pandas dataframe

dataset = pd.read\_csv(url, names=names)

dataset.head()

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 4].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors=5)

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

from sklearn.metrics import classification\_report, confusion\_matrix

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

error = []

# Calculating error for K values between 1 and 40

for i in range(1, 40):

knn = KNeighborsClassifier(n\_neighbors=i)

knn.fit(X\_train, y\_train)

pred\_i = knn.predict(X\_test)

error.append(np.mean(pred\_i != y\_test))

plt.figure(figsize=(12, 6))

plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',markerfacecolor='blue', markersize=10)

plt.title('Error Rate K Value')

plt.xlabel('K Value')

plt.ylabel('Mean Error')

**K-means Clustering**

**Write a program in python to perform k-means clustering on randomly generated data (150 samples, 2 samples) into 3 clusters. Label the centroids and plot the clusters.**

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

# create dataset

X, y = make\_blobs(

n\_samples=150, n\_features=2,

centers=3, cluster\_std=0.5,

shuffle=True, random\_state=0

)

# plot

plt.scatter(

X[:, 0], X[:, 1],

c='white', marker='o',

edgecolor='black', s=50

)

plt.show()

from sklearn.cluster import KMeans

km = KMeans(

n\_clusters=3, init='random',

n\_init=10, max\_iter=300,

tol=1e-04, random\_state=0

)

y\_km = km.fit\_predict(X)

# plot the 3 clusters

plt.scatter(

X[y\_km == 0, 0], X[y\_km == 0, 1],

s=50, c='lightgreen',

marker='s', edgecolor='black',

label='cluster 1'

)

plt.scatter(

X[y\_km == 1, 0], X[y\_km == 1, 1],

s=50, c='orange',

marker='o', edgecolor='black',

label='cluster 2'

)

plt.scatter(

X[y\_km == 2, 0], X[y\_km == 2, 1],

s=50, c='lightblue',

marker='v', edgecolor='black',

label='cluster 3'

)

# plot the centroids

plt.scatter(

km.cluster\_centers\_[:, 0], km.cluster\_centers\_[:, 1],

s=250, marker='\*',

c='red', edgecolor='black',

label='centroids'

)

plt.legend(scatterpoints=1)

plt.grid()

plt.show()